# Балансировка нагрузки в мультиэксабайтном сторадже



Вадим Зотеев,
старший разработчик






# Сторадж Яндекса

- Мульти-эксабайтный object storage
- Внутренние клиенты: Диск, Почта, Видео, Карты, Маркет и другие
- В основном cold и warm данные – hdd-диски
- Распределен в нескольких ДЦ
- Есть и репликация, и Erasure Coding





# 1

# Постановка задачи

# Постановка задачи

Строим распределенное отказоустойчивое хранилище

**write(key, value)**

**read(key) -> value**

# Постановка задачи

Строим распределенное отказоустойчивое хранилище

**write(key, value)**

< 300 ms

**read(key) -> value**

<  300 ms

# Постановка задачи

Строим распределенное отказоустойчивое хранилище

**write(key, value)**

< 300 ms

**read(key) -> value**

<  300 ms

Задача: балансировать write- и read-нагрузку **оптимальным образом**: не перегружая никакие диски и сеть

# Постановка задачи

Строим распределенное отказоустойчивое хранилище

**write(key, value)**

< 300 ms

**read(key) -> value**

<  300 ms

Задача: балансировать write- и read-нагрузку **оптимальным образом**: не перегружая никакие диски и сеть

**Должны переживать hardware-ошибки: от поломки диска до отключения ДЦ**

# Постановка задачи

Строим распределенное отказоустойчивое хранилище

**write(key, value)** — < 300 ms

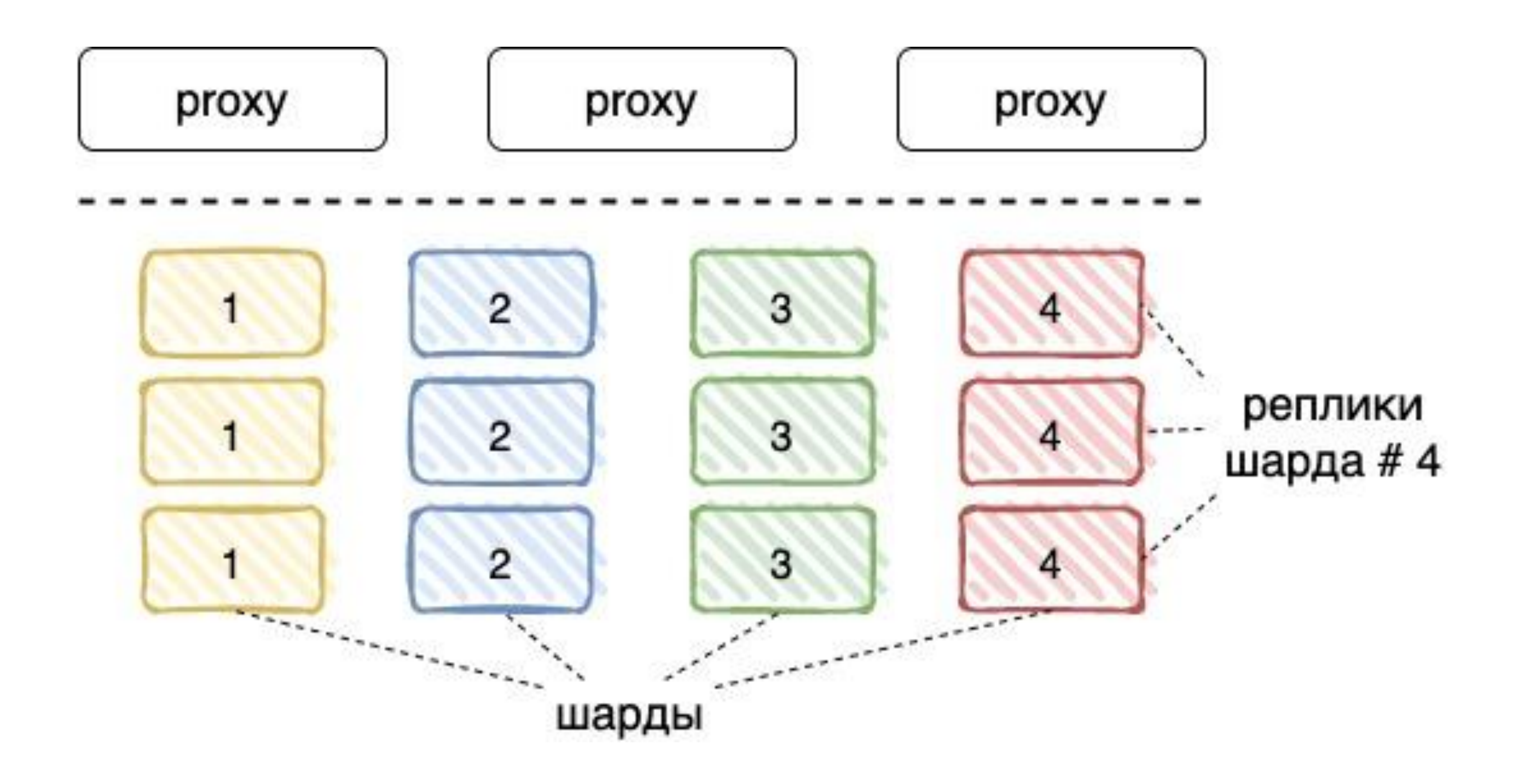
**read(key) -> value** — <  300 ms

Задача: балансировать write- и read-нагрузку **оптимальным образом**: не перегружая никакие диски и сеть

**Должны переживать hardware-ошибки: от поломки диска до отключения ДЦ**

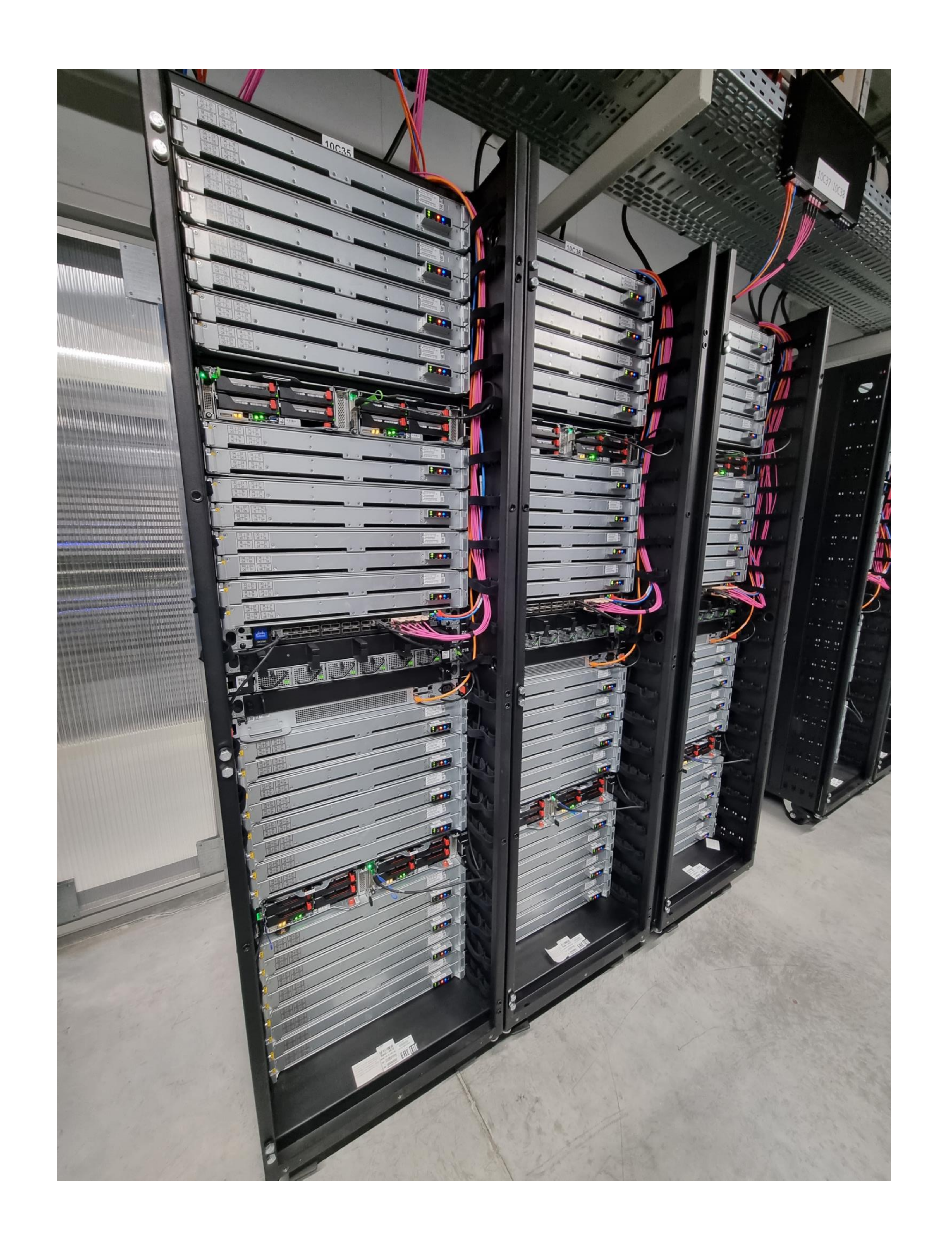
**Чем дешевле, тем лучше**

# Постановка задачи



# Постановка задачи

- Шарды фиксированного размера
- Диски «нарезаем» под шарды
- В каждом сервере много дисков



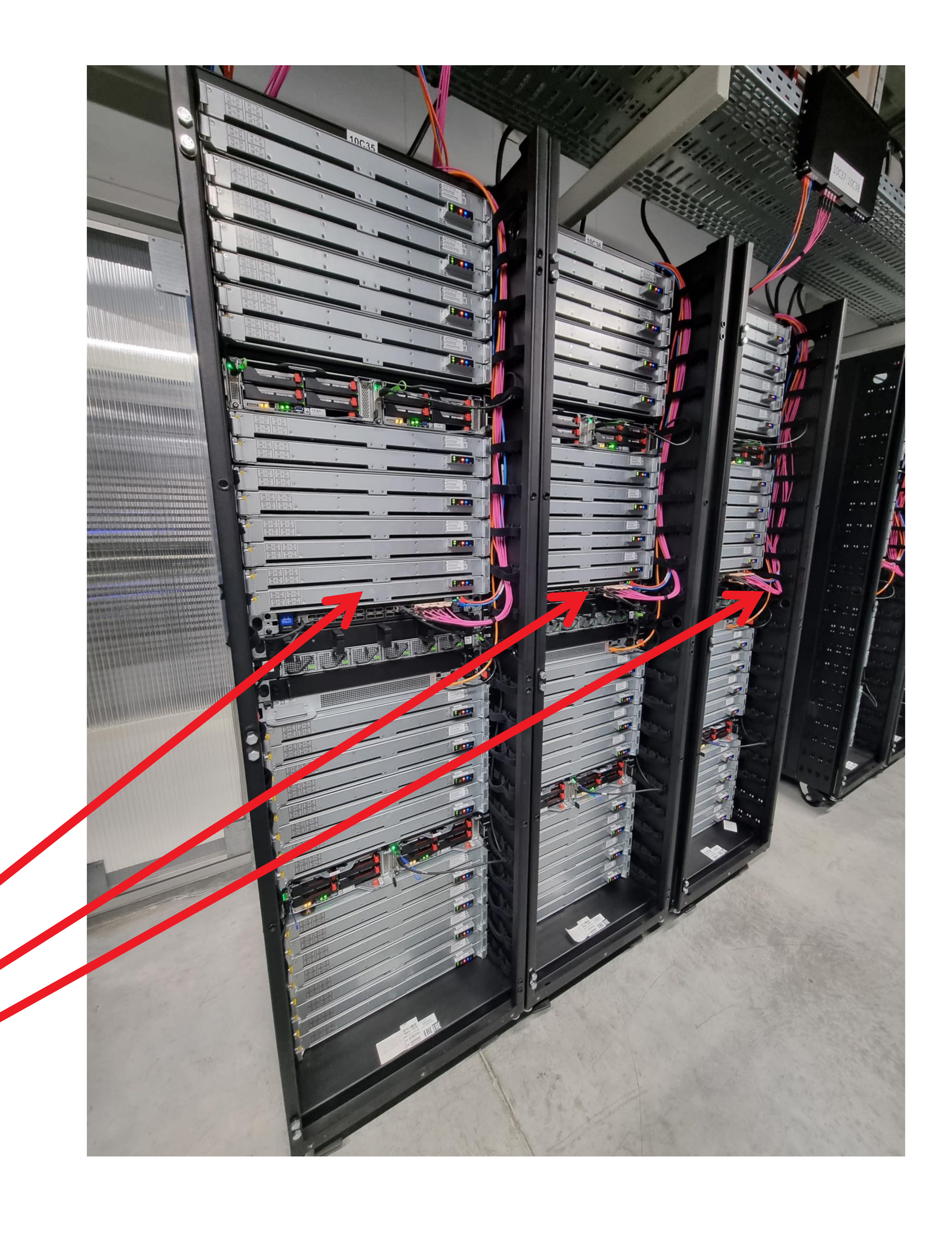






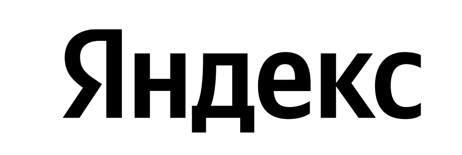

# Постановка задачи

+ Шарды фиксированного размера
+ Диски «нарезаем» под шарды
+ В каждом сервере много дисков



стойки








# Постановка задачи

- Шарды фиксированного размера
- Диски «нарезаем» под шарды
- В каждом сервере много дисков

4 сервера в стойке







# Постановка задачи

+ Шарды фиксированного размера
+ Диски «нарезаем» под шарды
+ В каждом сервере много дисков

по 100+ дисков
на сервер







2

## Балансировка write-нагрузки

# Вариант распределения нагрузки по шардам: hashing

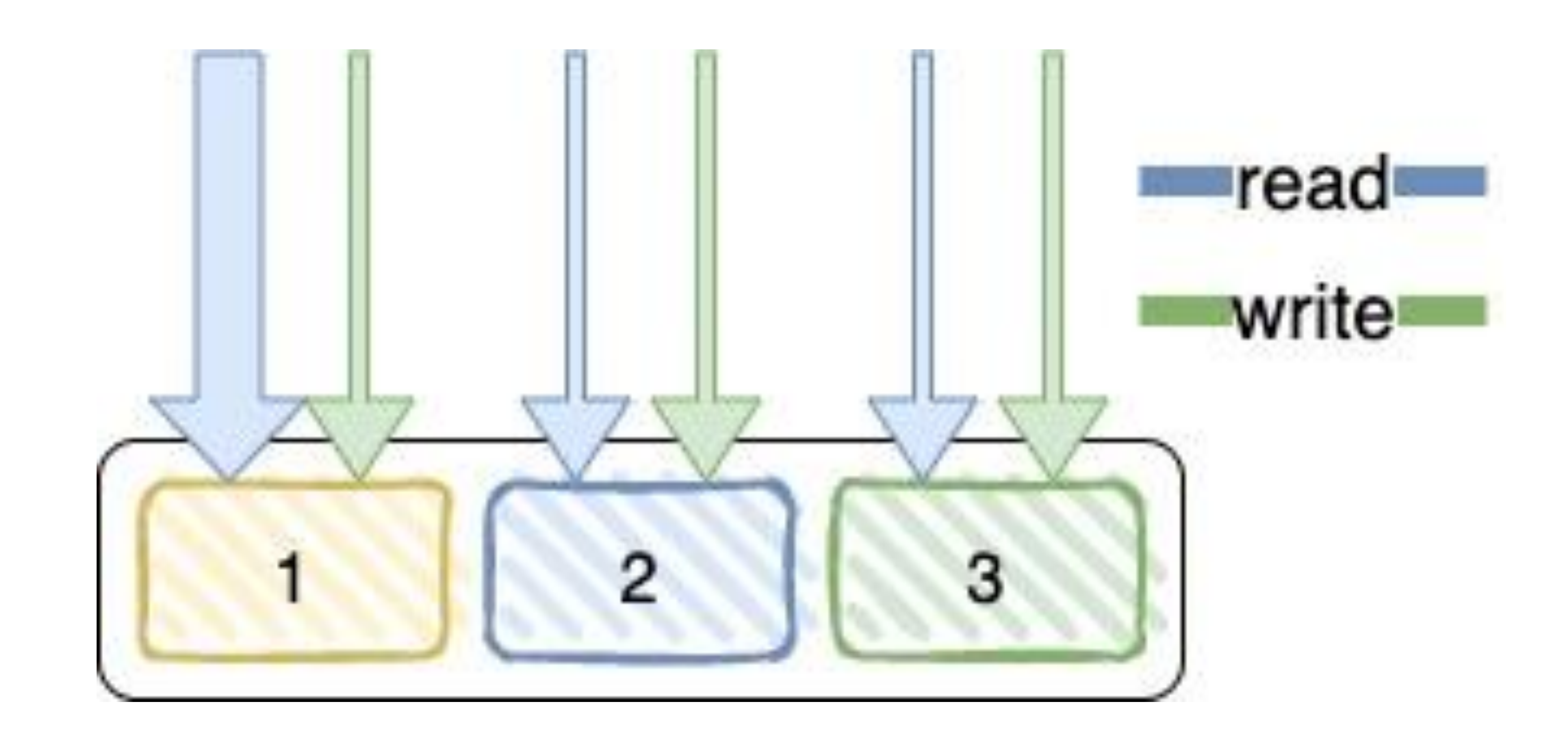
Детерминированный алгоритм записи:
+ hash(key) % N
+ Consistent hashing

Необходим переезд данных при изменении N

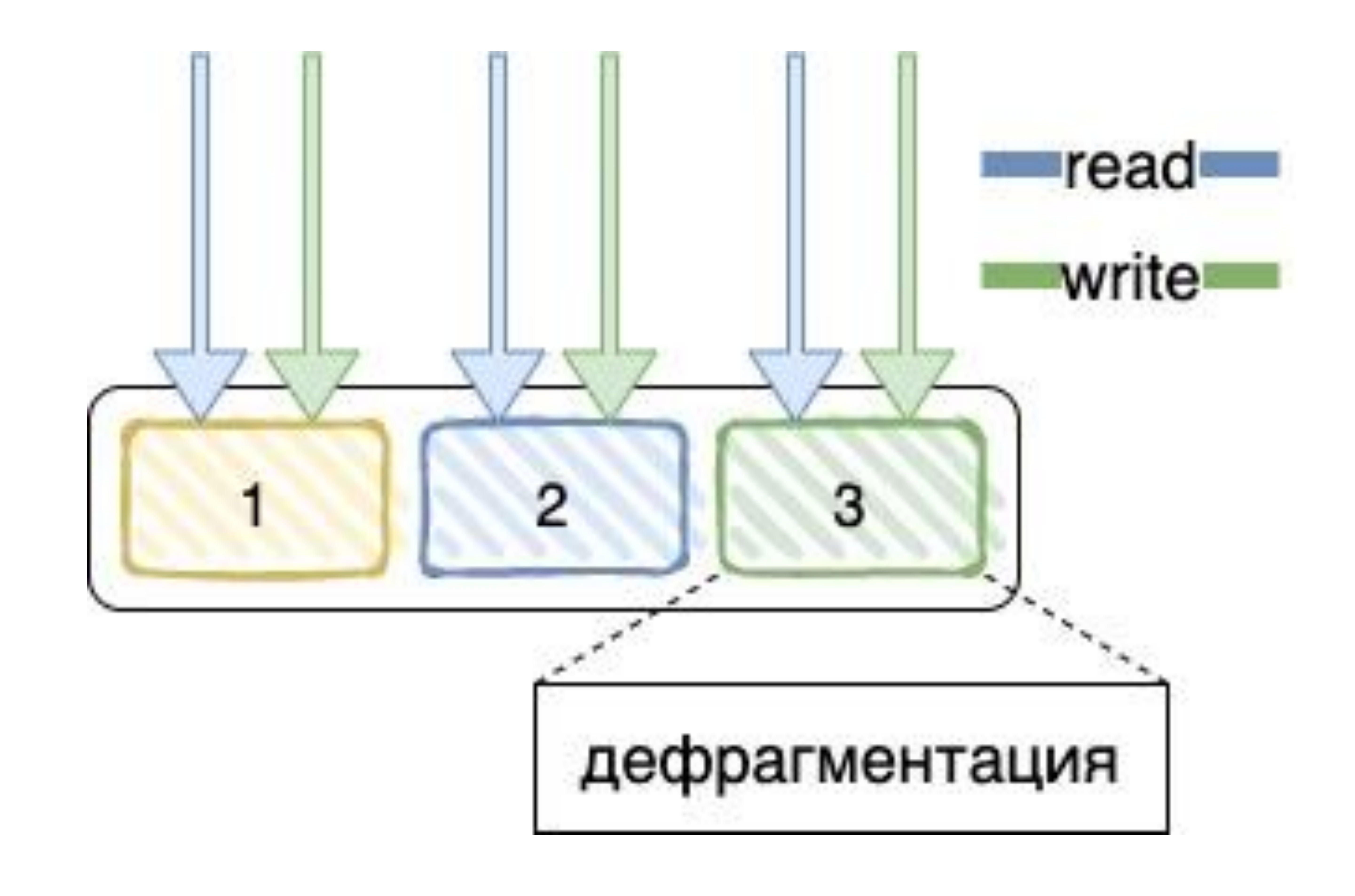




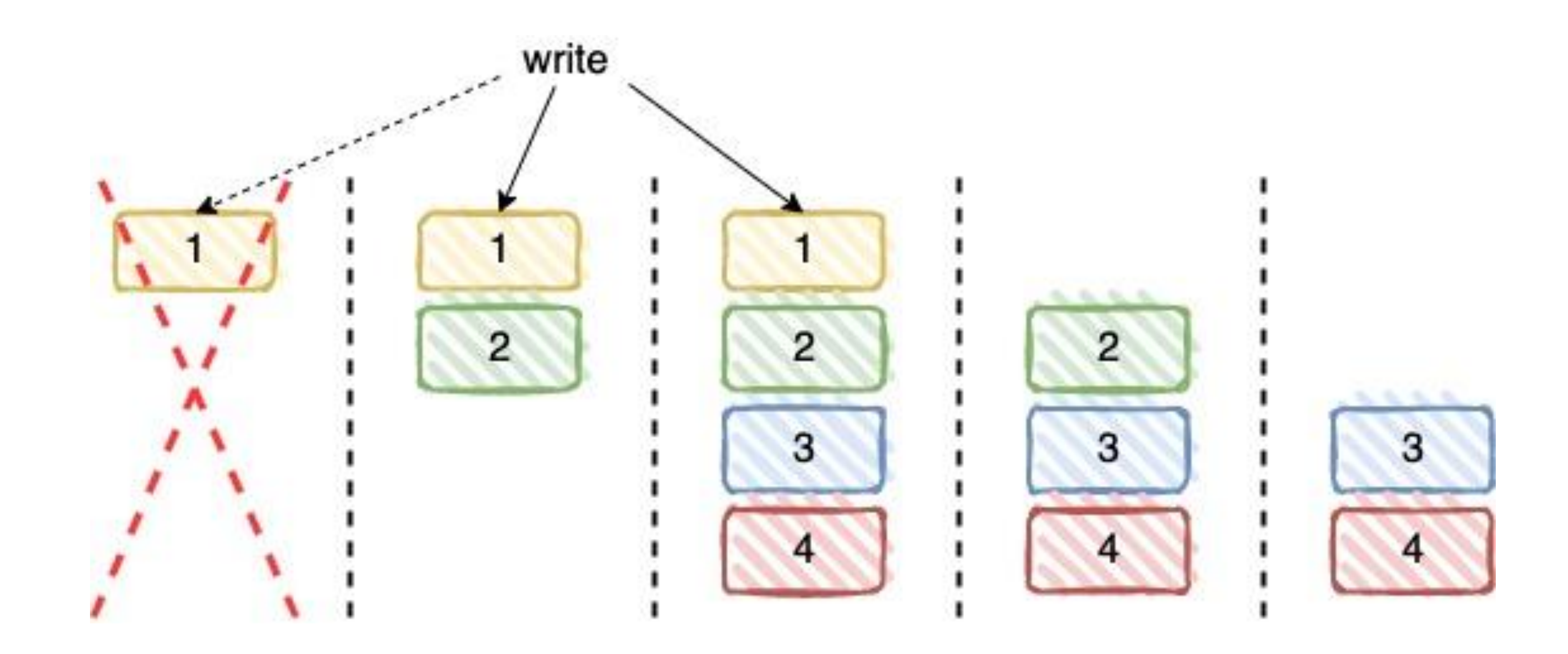
# Проблема: диск перегружен



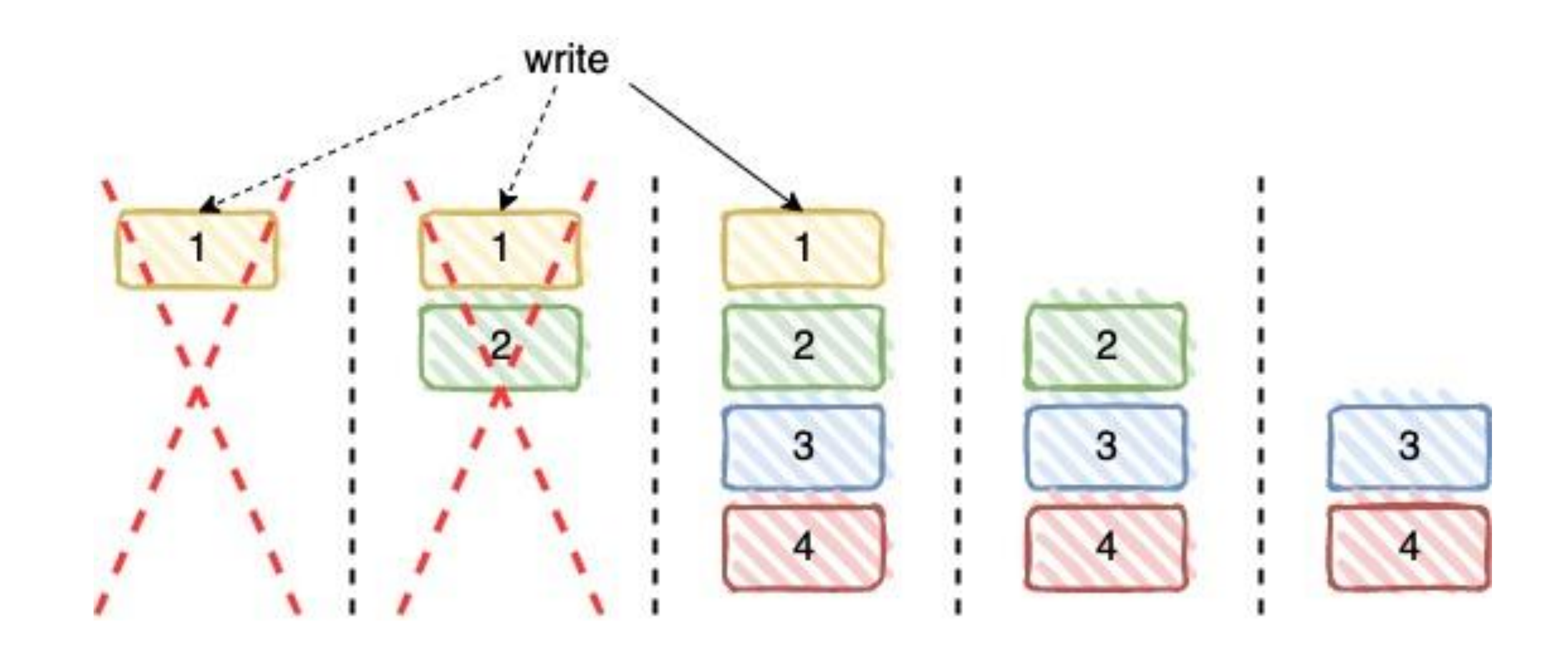
# Проблема: фоновый процесс



# Проблема: потеря реплики



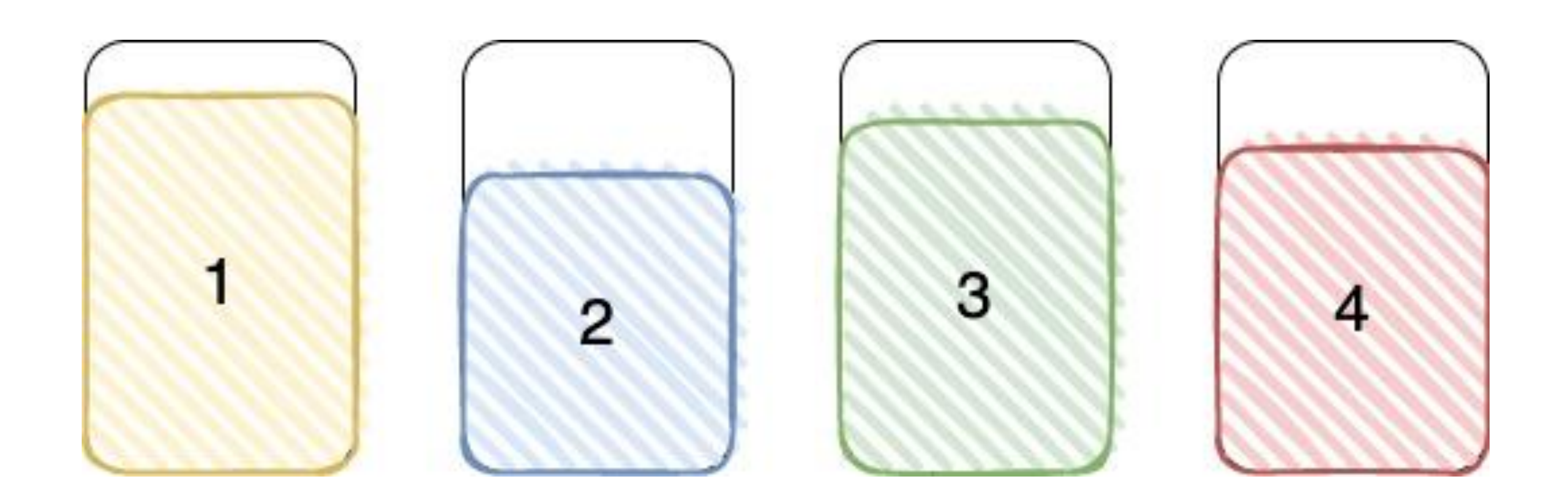
# Проблема: потеря реплики





Яндекс HL HighLoad++ 2022

# Проблема: неравномерное заполнение шардов




Данные разного размера -> шарды заполняются неравномерно

# Нет контроля над записью!

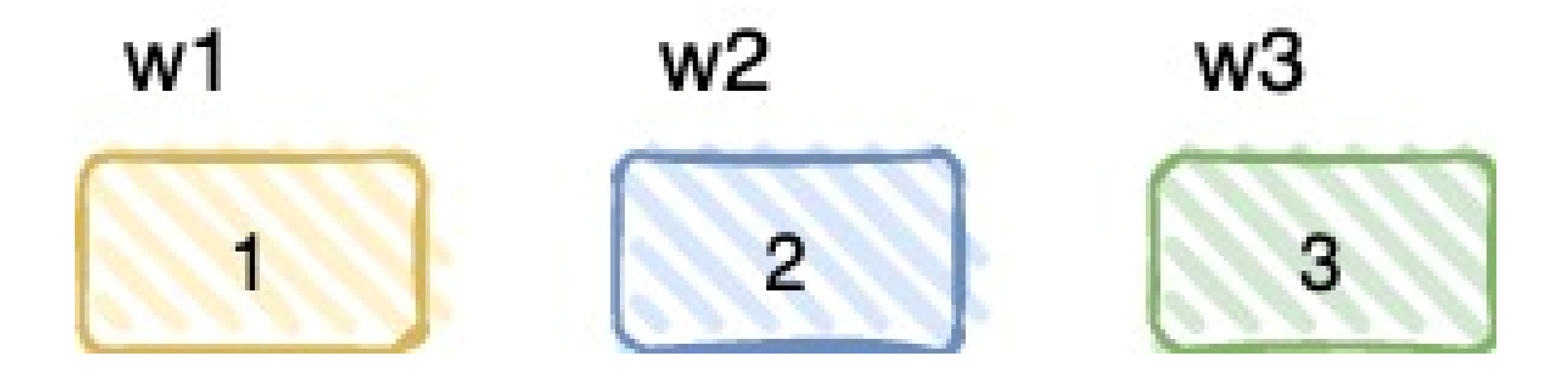
# Решение: управлять записью



# Решение: управлять записью

Выбор шарда при каждой записи

Недетерминированный алгоритм: взвешенный рандом



# Как считать веса

Вес = произведение коэффициентов





# Как считать веса

Вес = произведение коэффициентов

+ Утилизация диска

$$W = \frac{1}{U_{disk}}$$

# Как считать веса

Вес = произведение коэффициентов

- Утилизация диска
- Утилизация сети

$$W = \frac{1}{U_{disk} * \mathbf{U_{net}}}$$

# Как считать веса

Вес = произведение коэффициентов

+ Утилизация диска
+ Утилизация сети
+ % свободного места

$$W = \frac{U_{free_space}}{U_{disk} * U_{net}}$$

# Как считать веса

Вес = произведение коэффициентов

- Утилизация диска
- Утилизация сети
- % свободного места
- «Живость» всех реплик

$$W = \frac{U_{free_space}}{U_{disk} * U_{net}} * \mathbf{B}_{\mathbf{avail}}$$

# Как считать веса

Вес = произведение коэффициентов

- Утилизация диска
- Утилизация сети
- % свободного места
- «Живость» всех реплик
- Ваш вариант

$$W = \frac{U_{free_space}}{U_{disk} * U_{net}} * B_{avail} * \mathbf{F}$$

# Как считать веса

Вес = произведение коэффициентов

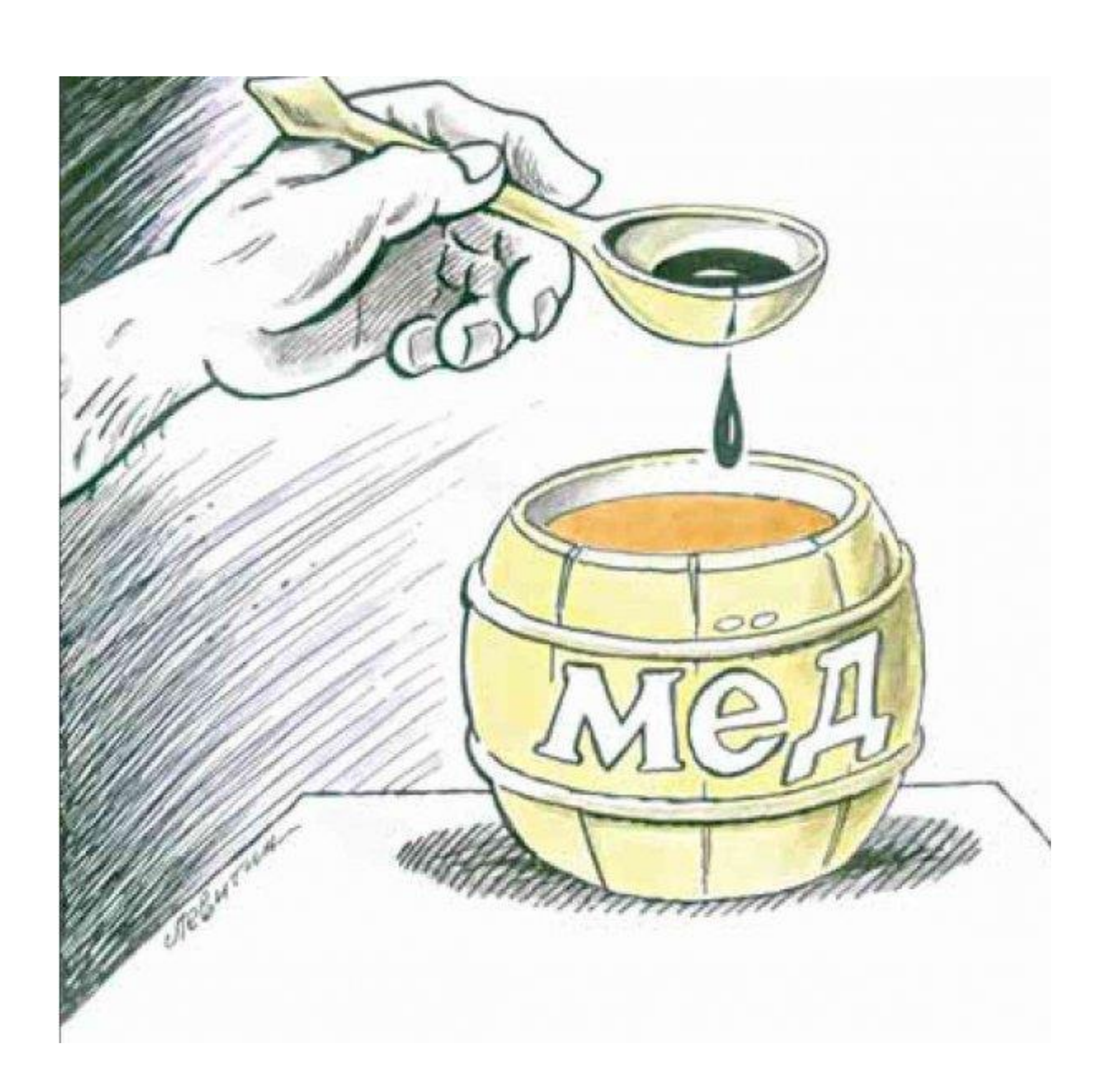
- Утилизация диска
- Утилизация сети
- % свободного места
- «Живость» всех реплик
- Ваш вариант

$$W = \frac{U_{free_space}}{U_{disk} * U_{net}} * B_{avail} * F$$

Можно считать для части шардов

# Нужно хранить id шарда

+ Нужна мета-база
+ Мета-база нагруженная



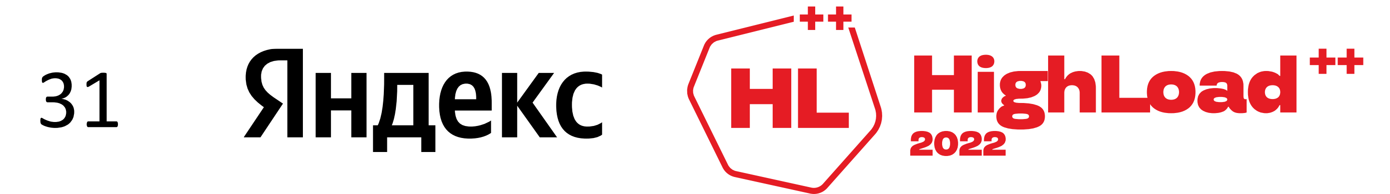
# Нужно хранить id шарда

- Нужна мета-база
- Мета-база нагруженная

Можно хранить на стороне клиента

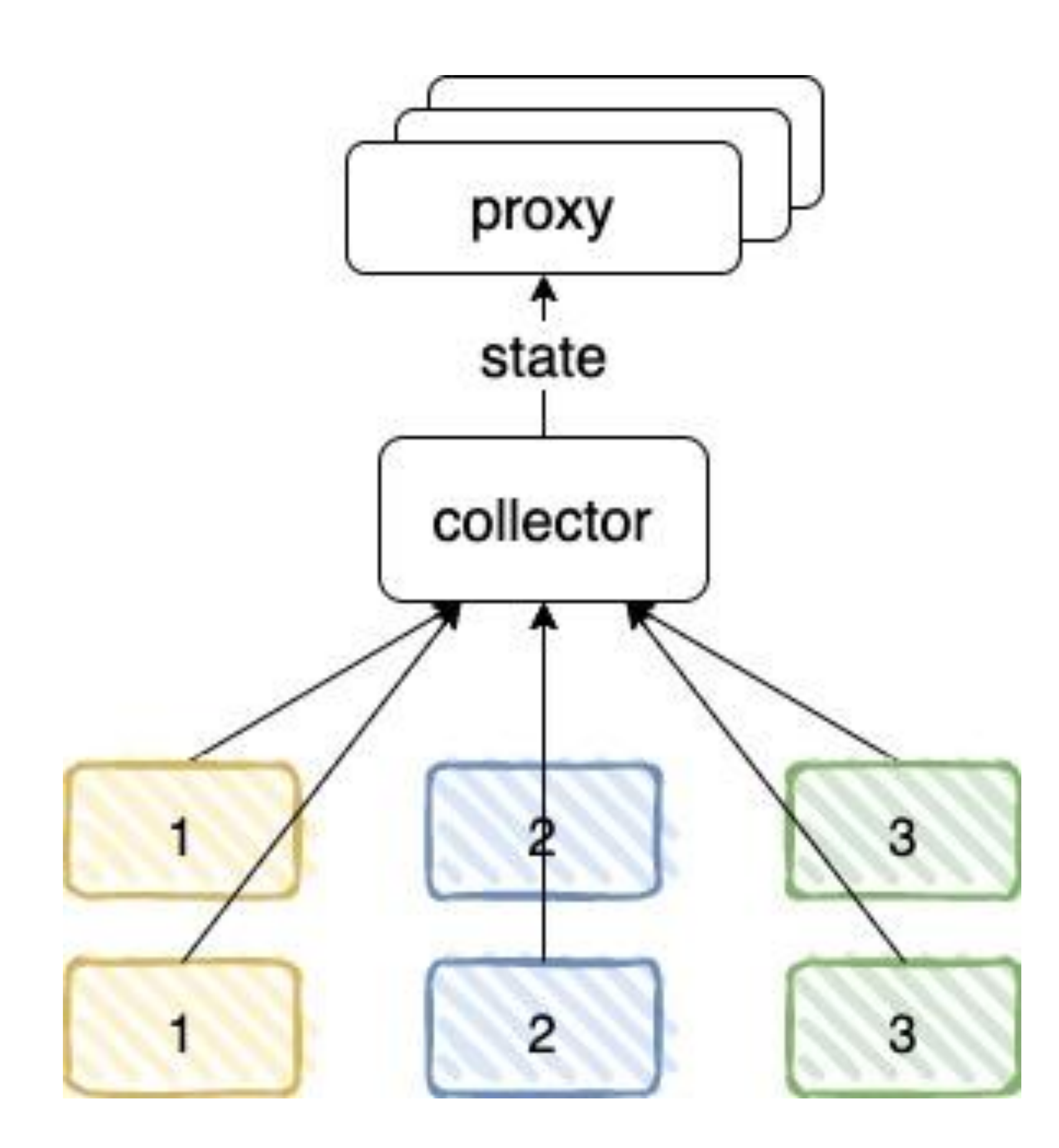




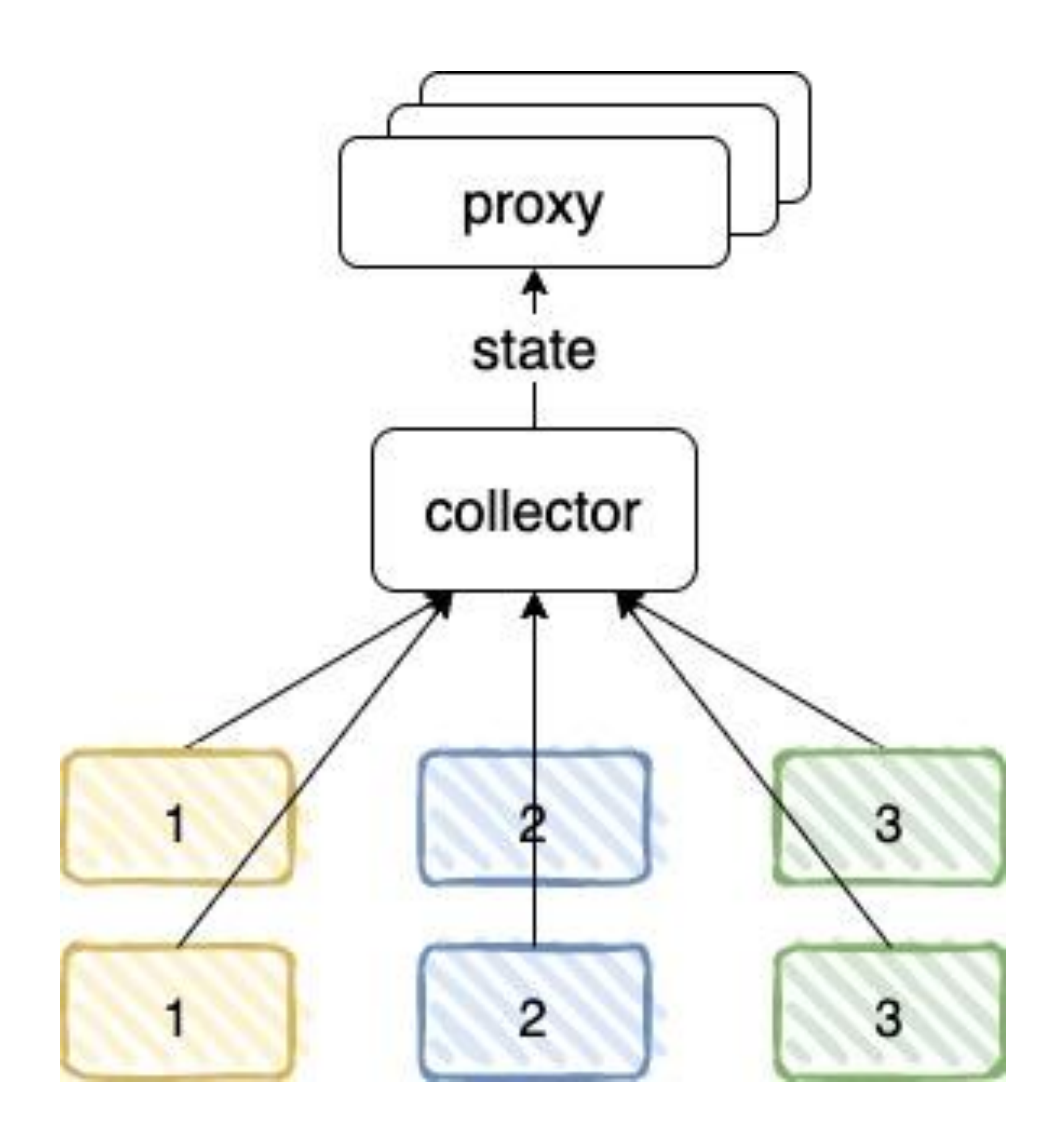
# Сбор стейта

+ Веса зависят от стейта



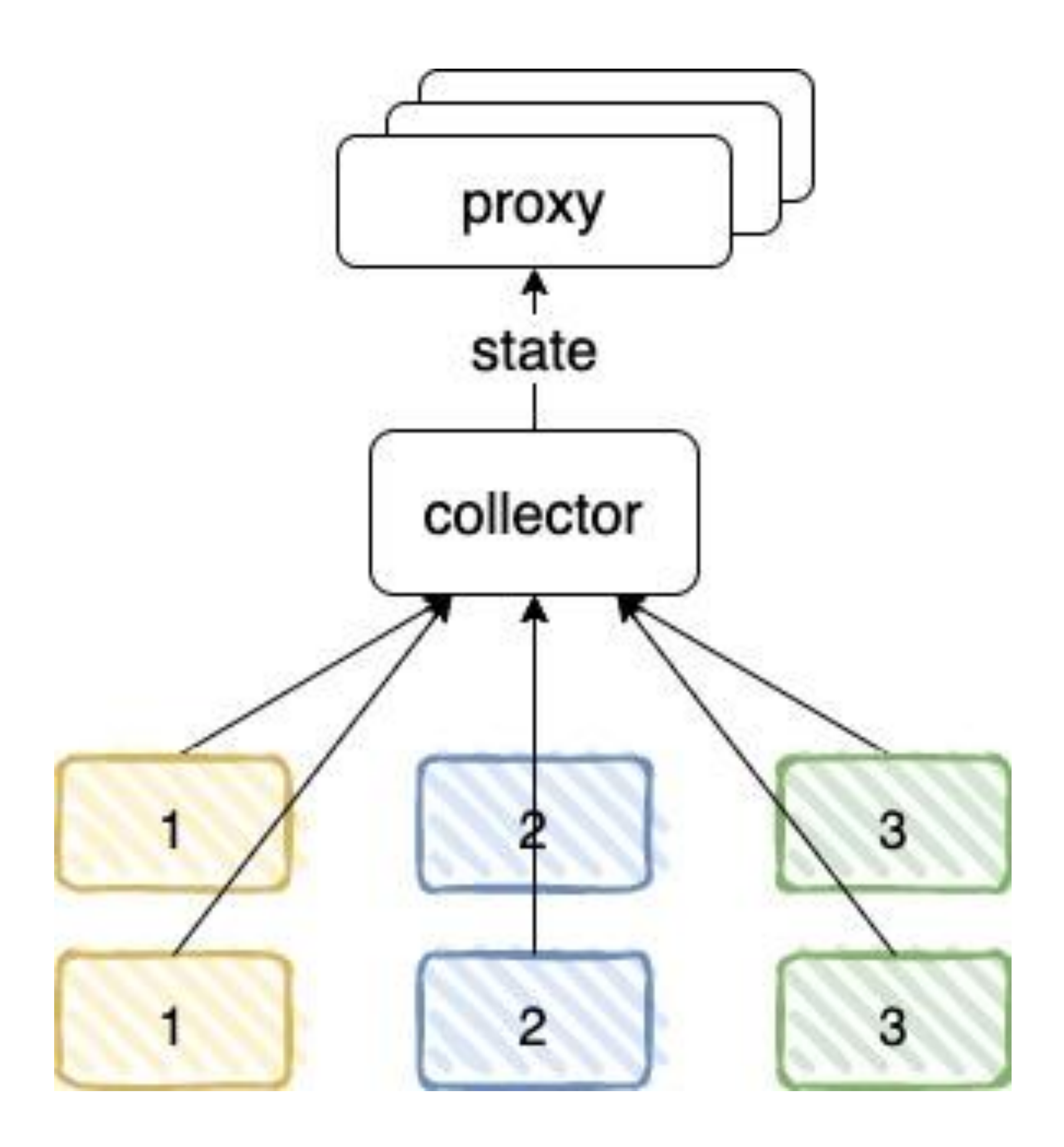
# Сбор стейта

- Веса зависят от стейта
- Стейт «варится» не мгновенно



# Сбор стейта

- Веса зависят от стейта
- Стейт «варится» не мгновенно
- Сборщик стейта – критичный компонент



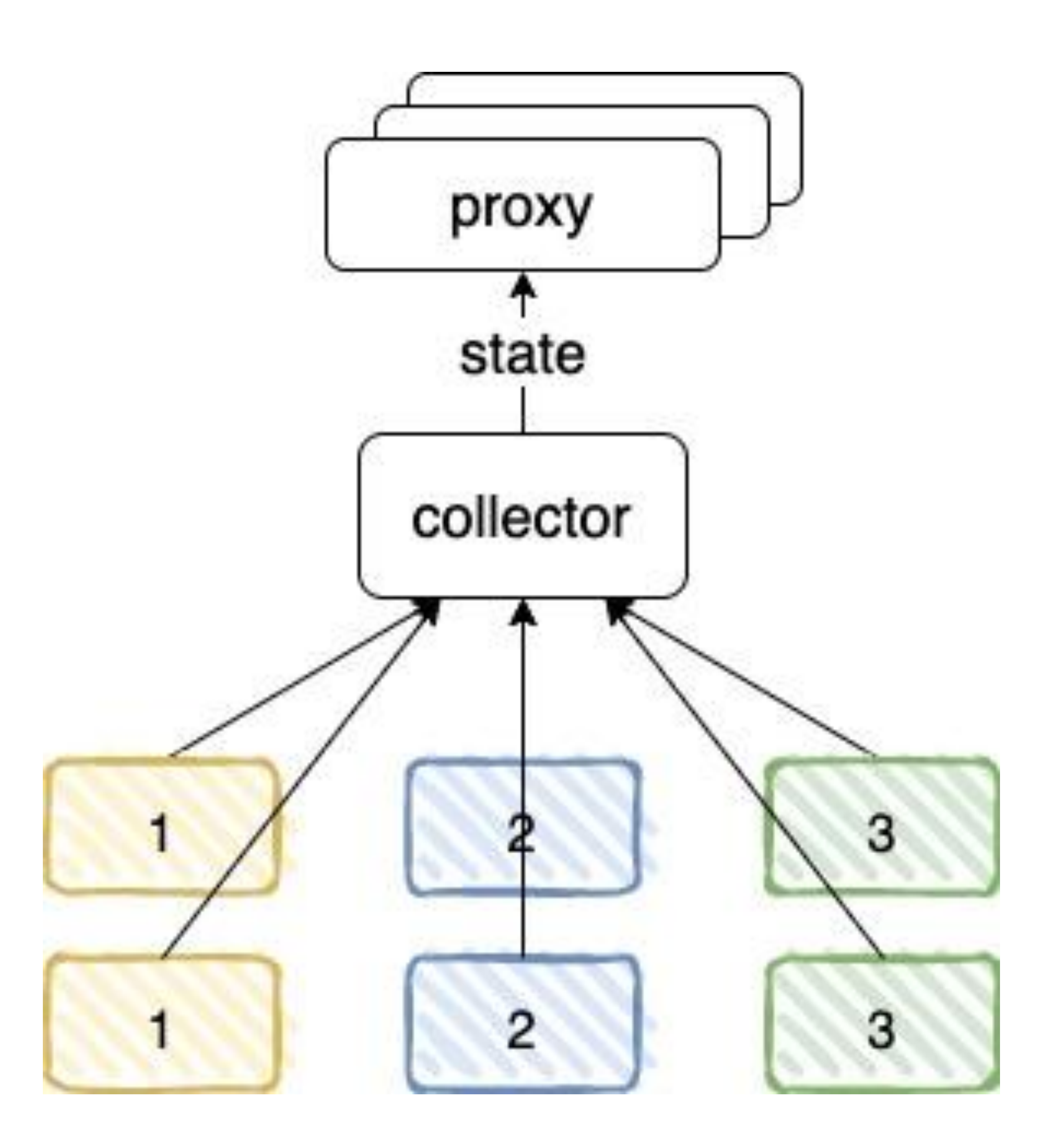





# Сбор стейта

- Веса зависят от стейта
- Стейт «варится» не мгновенно
- Сборщик стейта – критичный компонент

Между consistency и availability выбираем availability



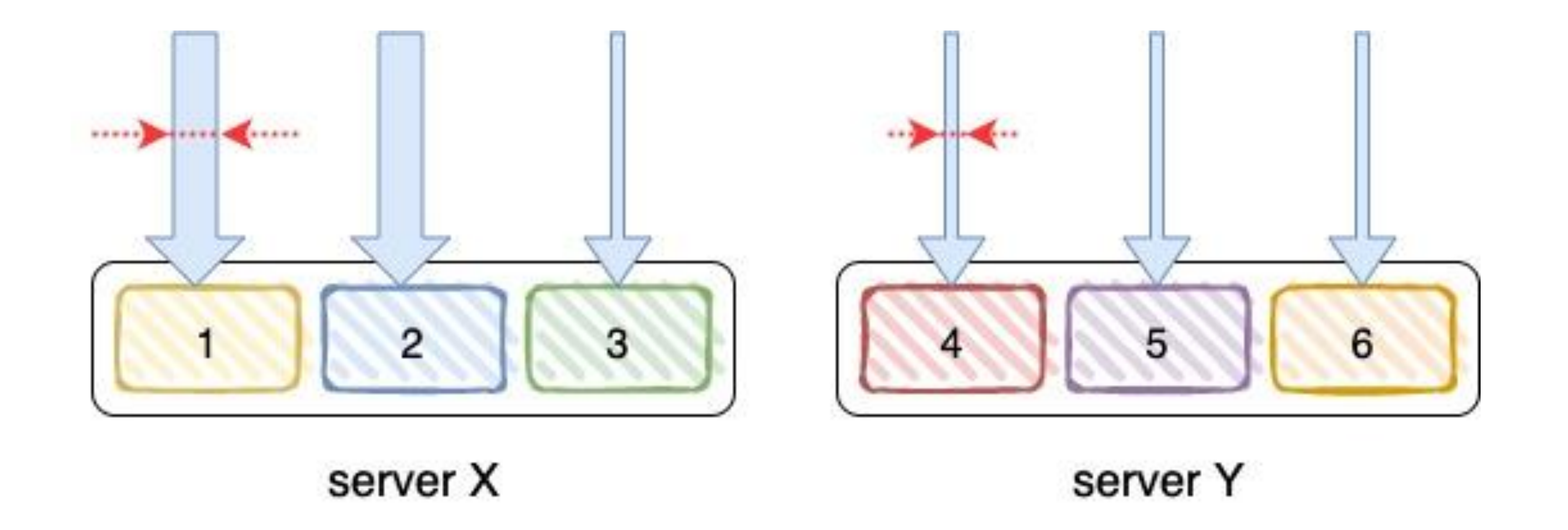





# 3

# Балансировка read-нагрузки

# Проблема: неравномерное распределение нагрузки

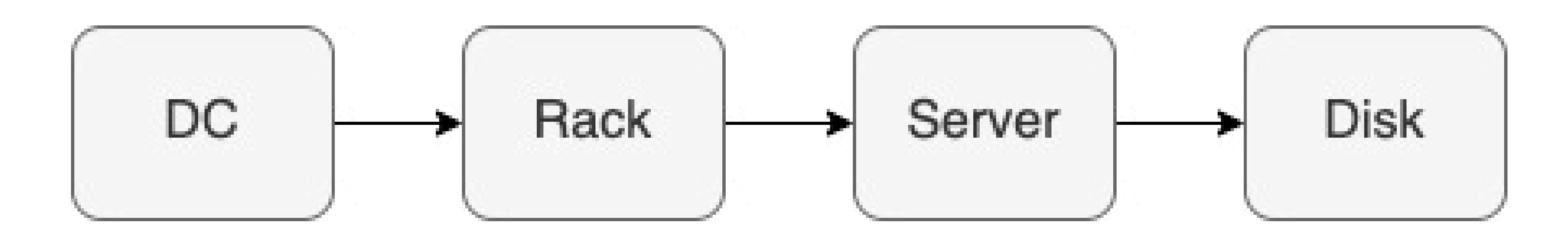


# Равномерное распределение шардов

- Шарды разных сервисов равномерно распределяем по железу
- «Горячие» шарды селим рядом с «холодными» (у разных сервисов разная нагрузка)

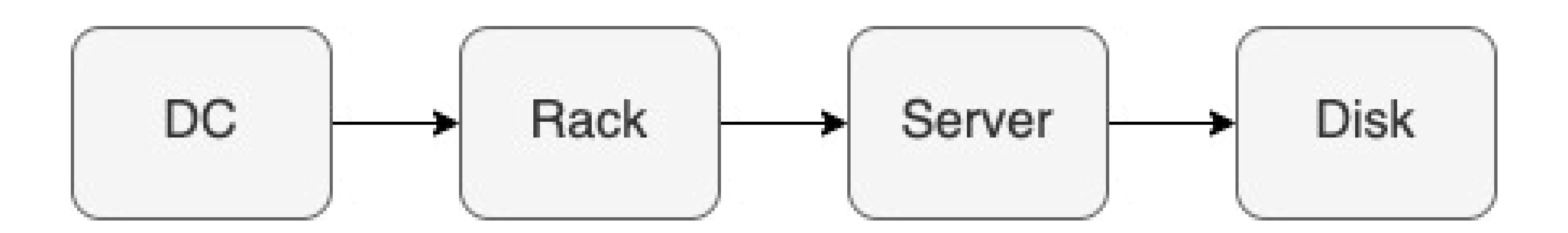
# Равномерное распределение шардов

- Шарды разных сервисов равномерно распределяем по железу
- «Горячие» шарды селим рядом с «холодными» (у разных сервисов разная нагрузка)
- Нас интересуют: диски, сервера, стойки и целые ДЦ



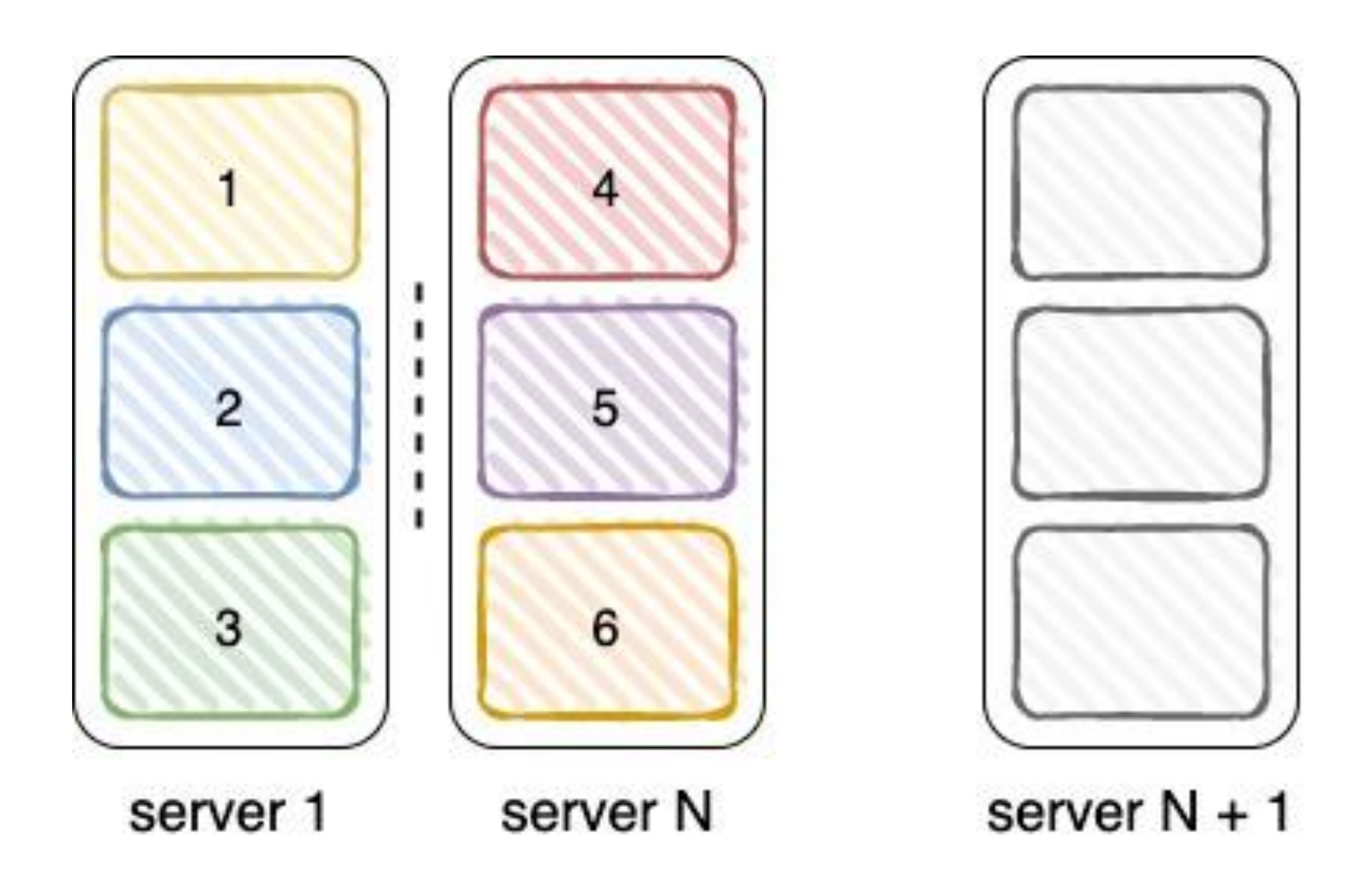
# Равномерное распределение шардов

- Шарды разных сервисов равномерно распределяем по железу
- «Горячие» шарды селим рядом с «холодными» (у разных сервисов разная нагрузка)
- Нас интересуют: диски, сервера, стойки и целые ДЦ
- Помогает распределять write-нагрузку



# Проблема: добавление нового железа

- Сначала высокая write-нагрузка
- Потом высокая read-нагрузка

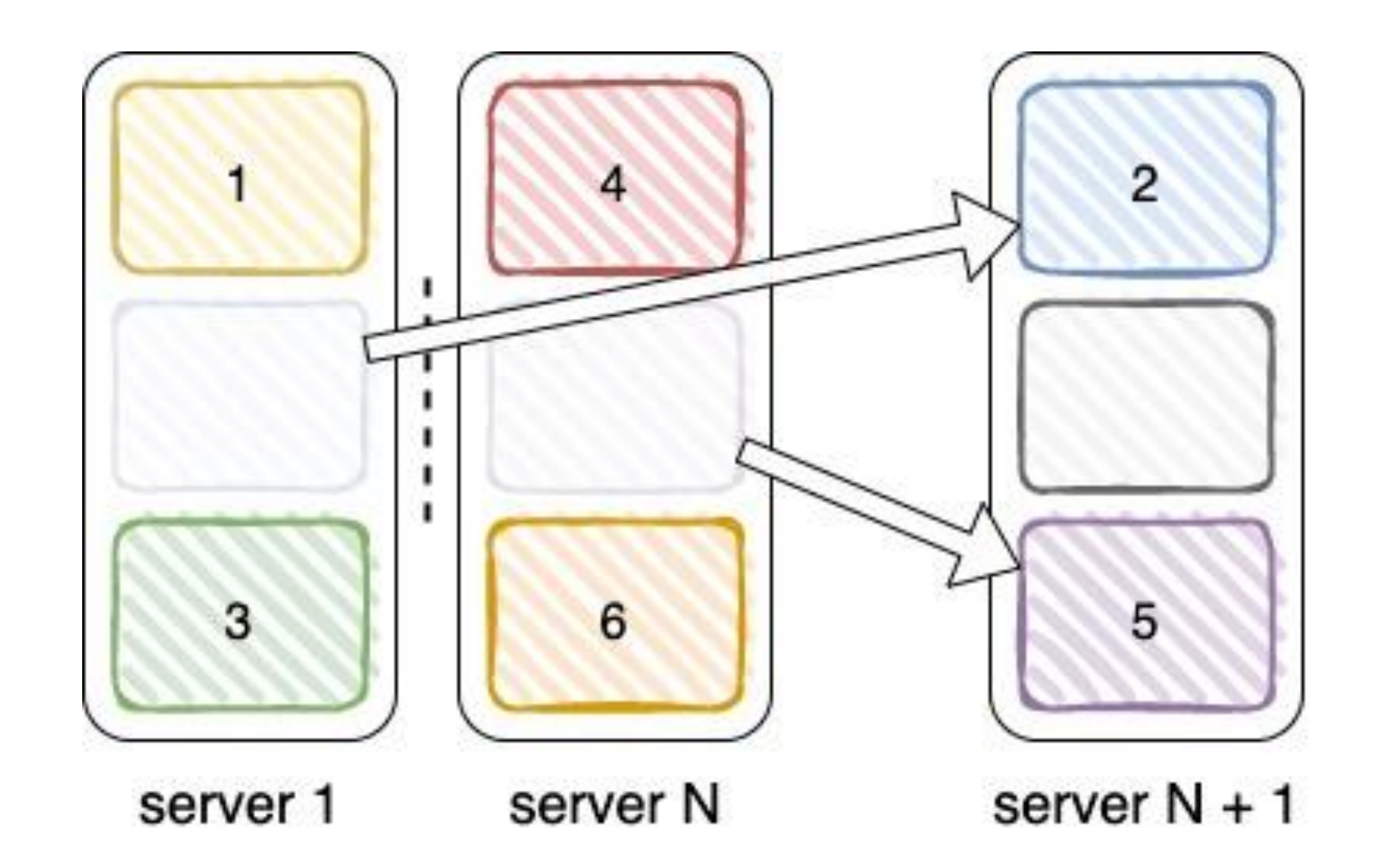




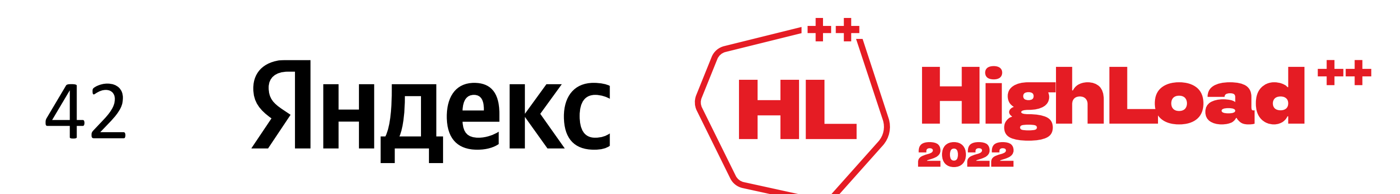



# Фоновый переезд данных

- Равномерно распределяем свободное место
- Простая стратегия: % пустых шардов

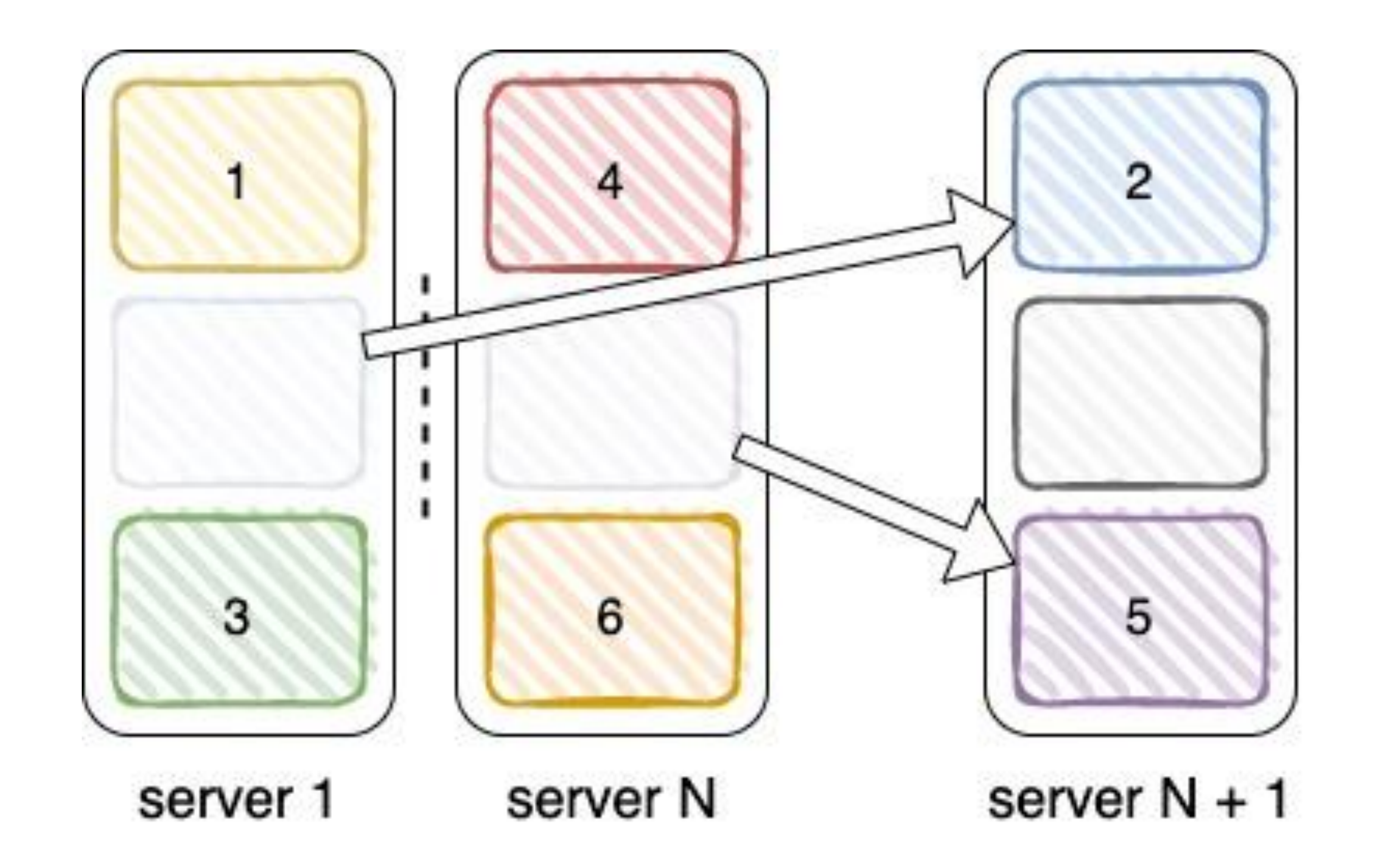




# Фоновый переезд данных

- Равномерно распределяем свободное место
- Простая стратегия: % пустых шардов
- Нужен шедулинг и процессинг переездов



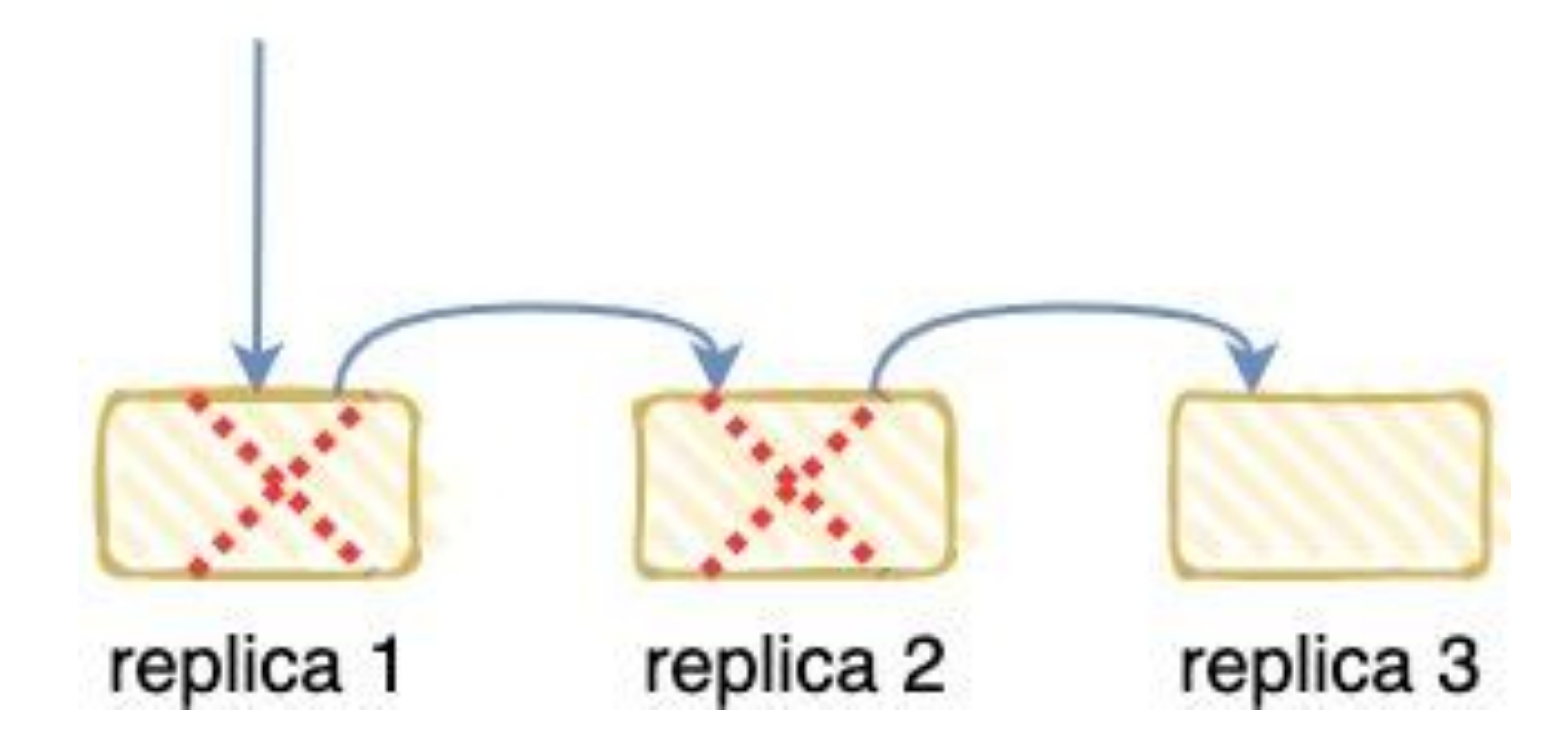





# Как выбрать реплику?

# Выбор реплики

- Sequential

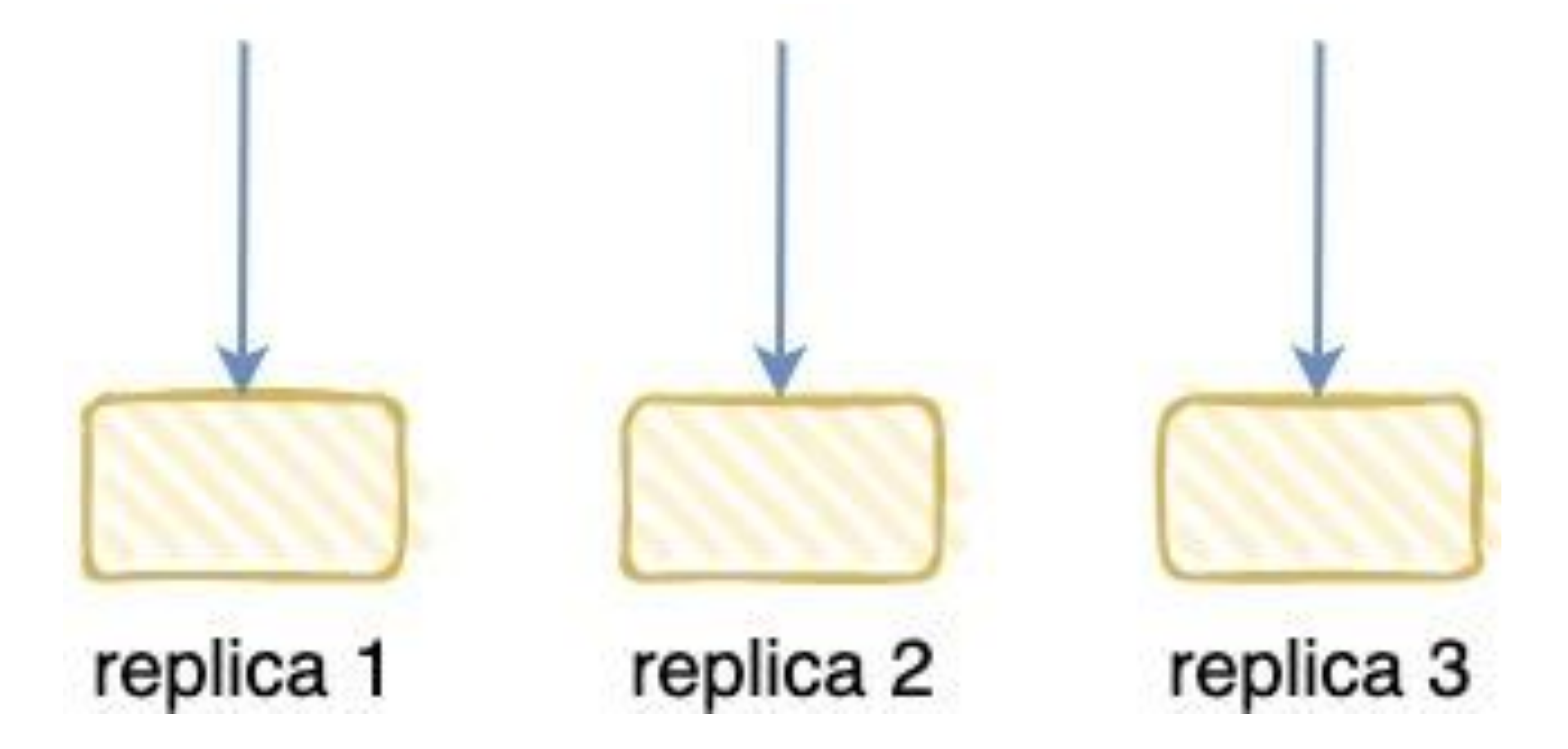






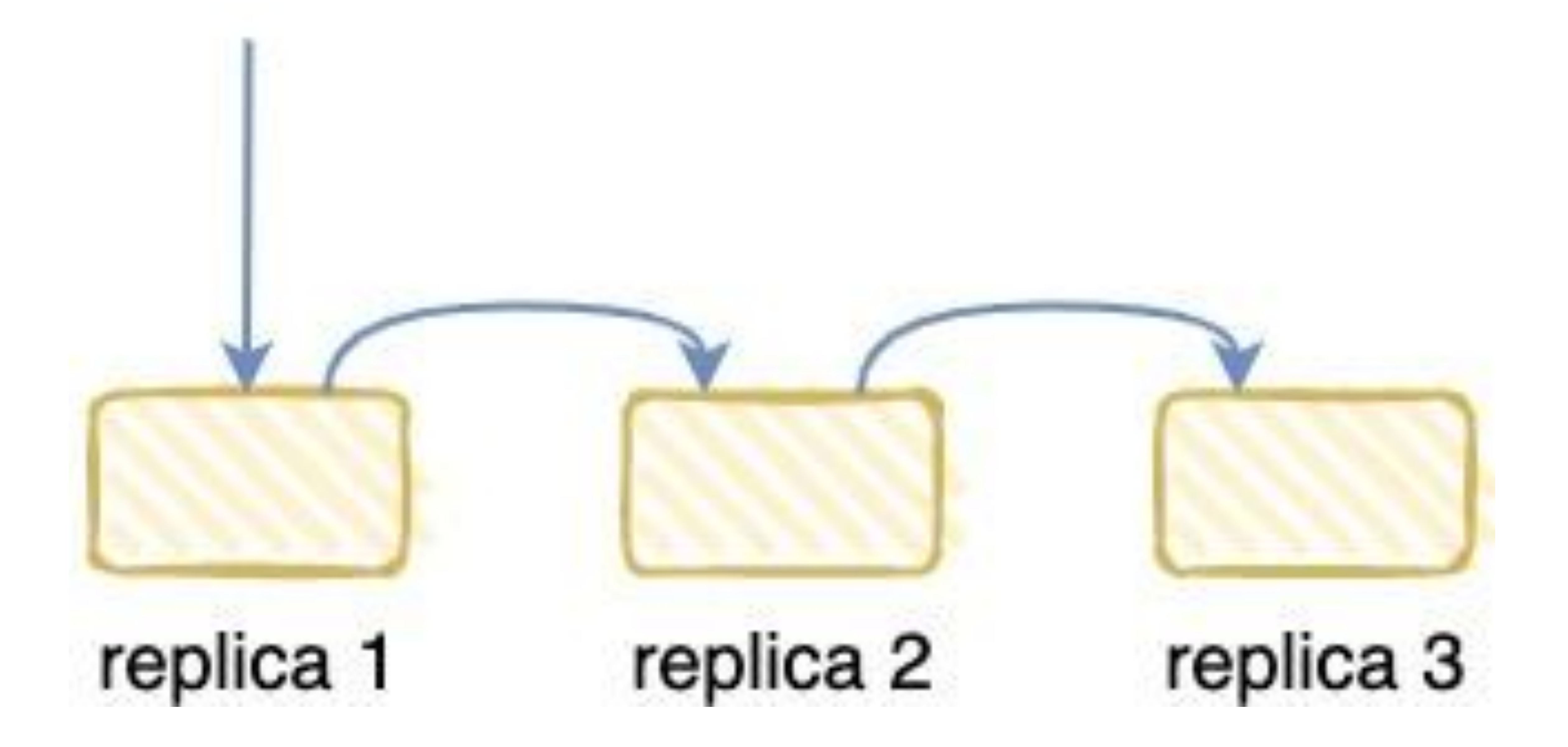

# Выбор реплики

- Sequential
- **Parallel**



# Выбор реплики

- Sequential
- Parallel
- **Soft sequential**





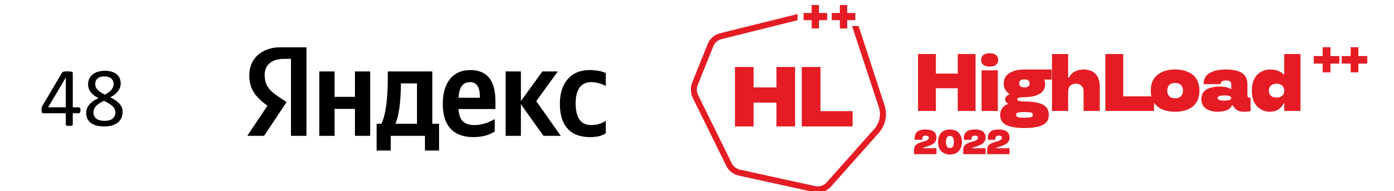



# Выбор реплики

- Sequential
- Parallel
- Soft sequential

Подходящая стратегия зависит от сервиса: нагрузки, среднего размера данных

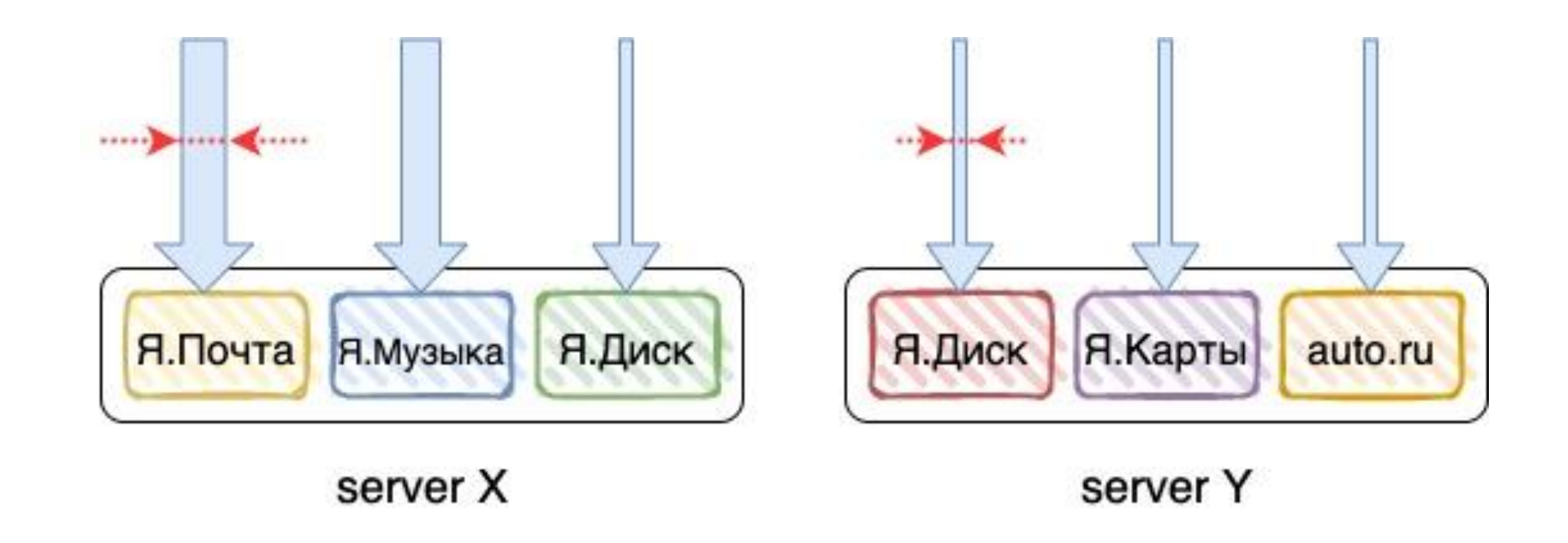




# 4

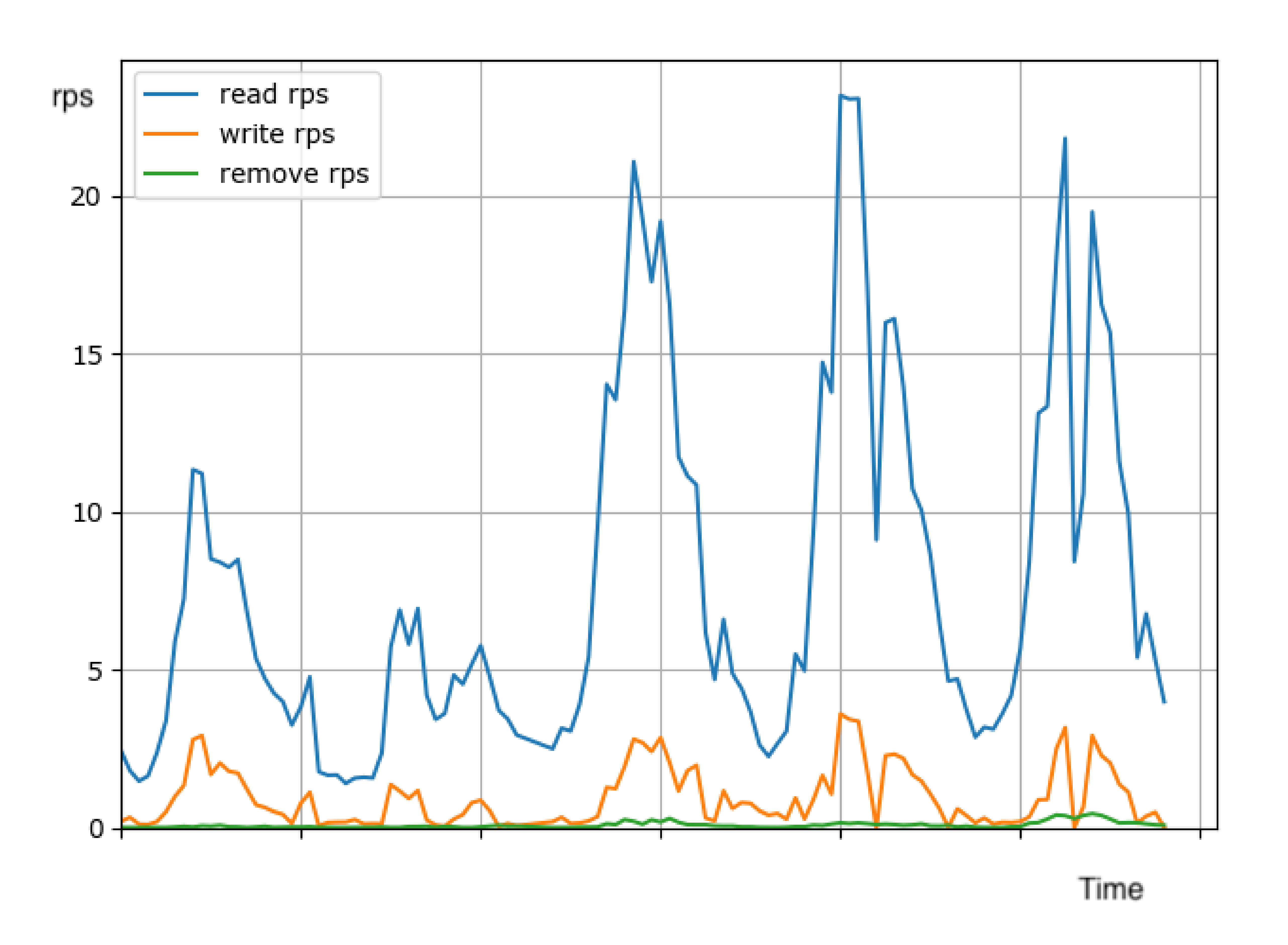
# Предсказание нагрузки

# Проблема: «горячие» шарды разных сервисов рядом



# Считаем нагрузку

- Предсказание нагрузки – нужен анализ логов
- Оценка будущей нагрузки – скользящее среднее

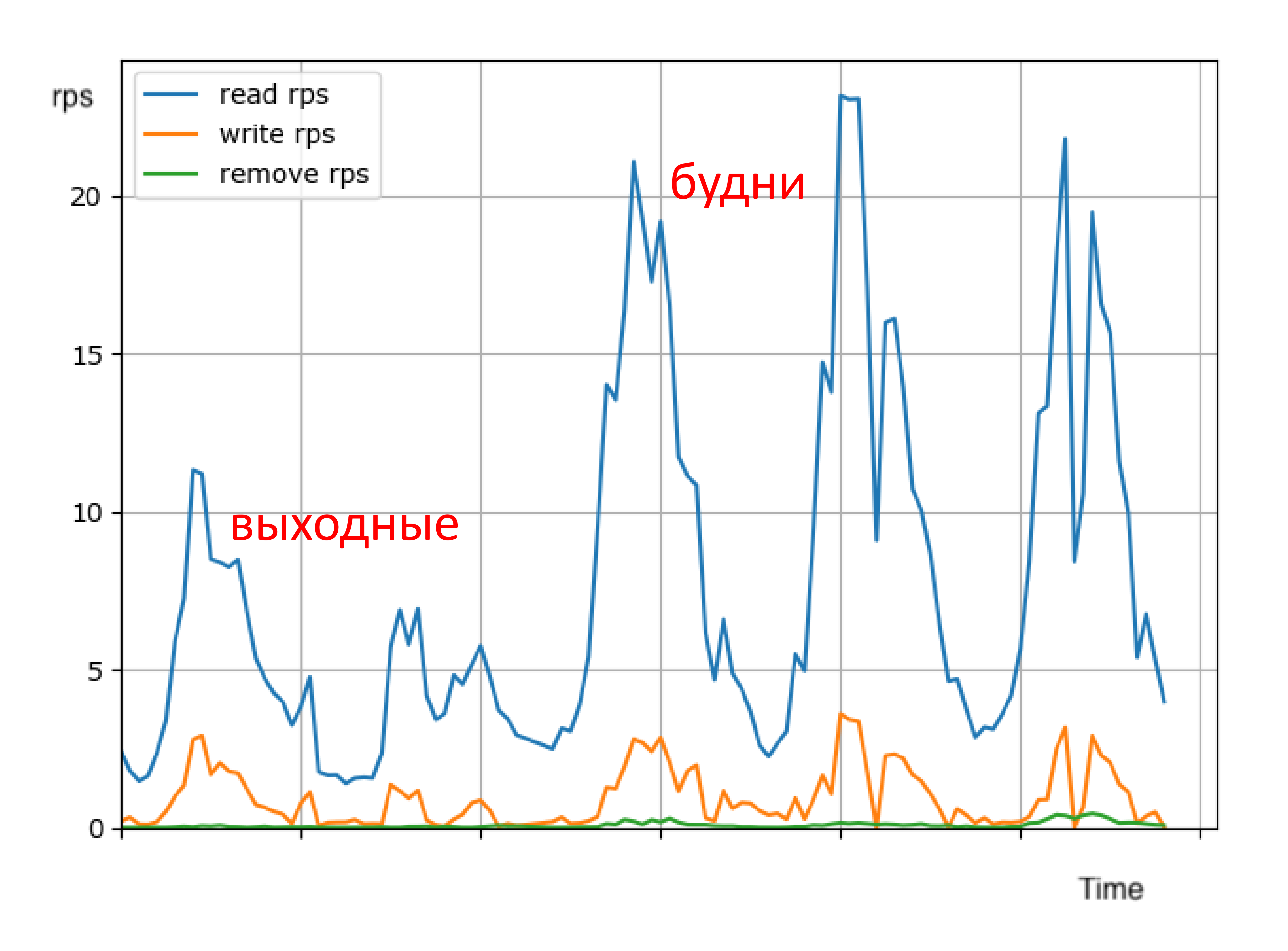







# Считаем нагрузку

+ Предсказание нагрузки – нужен анализ логов
+ Оценка будущей нагрузки – скользящее среднее



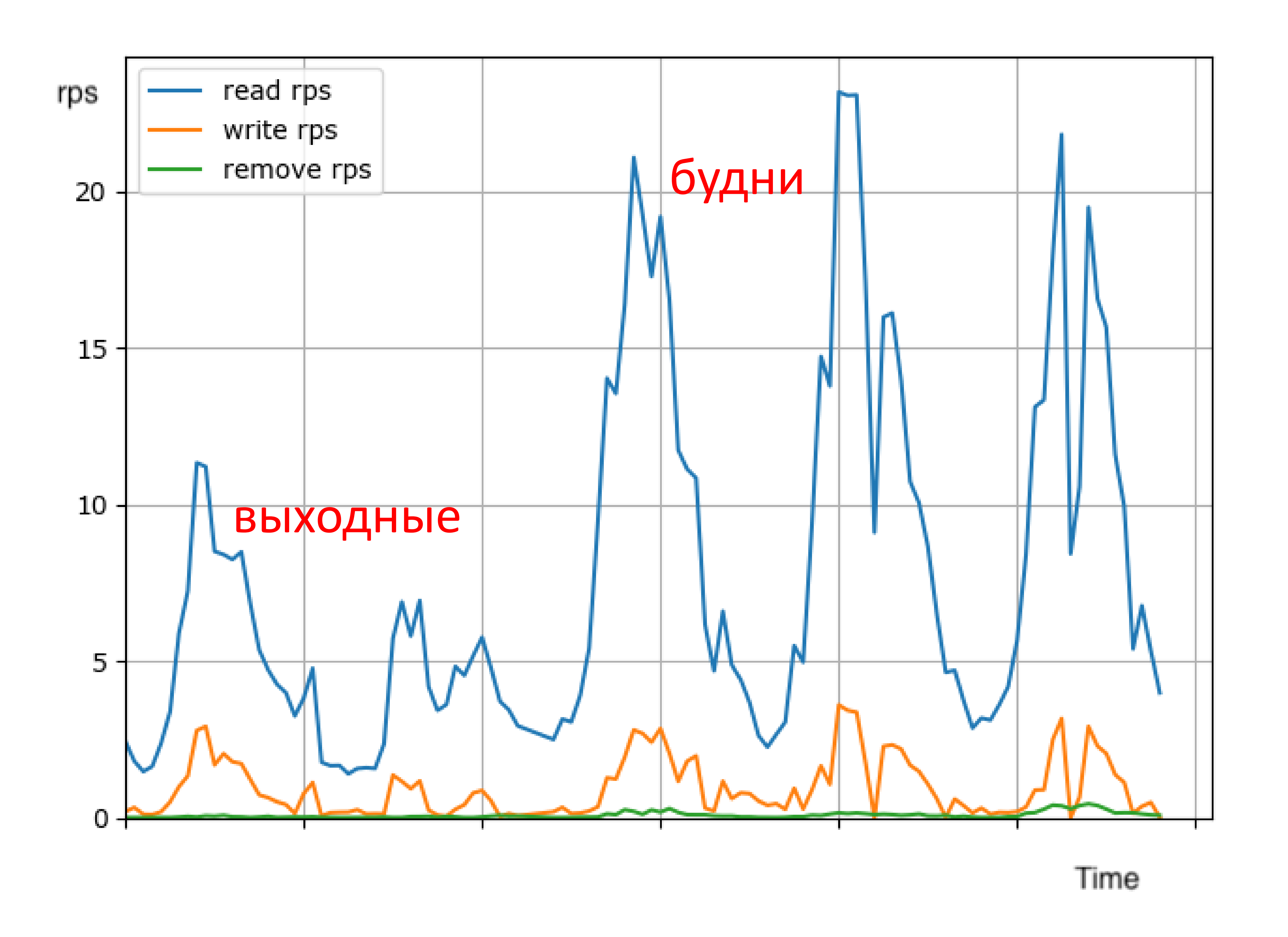





# Считаем нагрузку

- Предсказание нагрузки – нужен анализ логов
- Оценка будущей нагрузки – скользящее среднее

Считать iops’ы непросто!



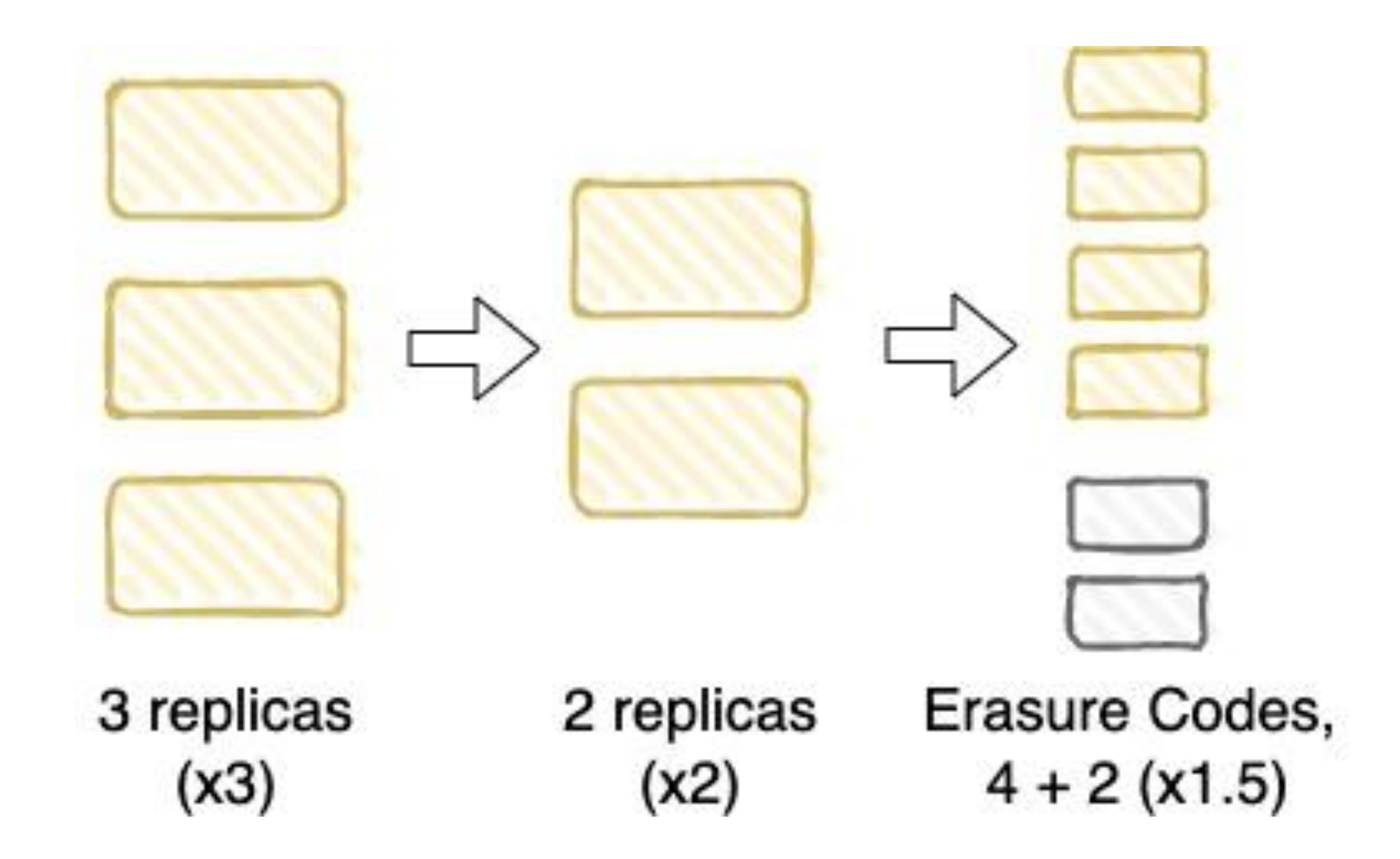





# Оптимальный способ хранения

Можем подобрать оптимальный способ хранения для шарда:

- Изменить число реплик
- Хранить с помощью Erasure Codes








# Выводы

В большом хранилище:

+ Нужен явный контроль за записью (это не бесплатно)





# Выводы

В большом хранилище:

+ Нужен явный контроль за записью (это не бесплатно)
+ Равномерное распределение шардов разных сервисов



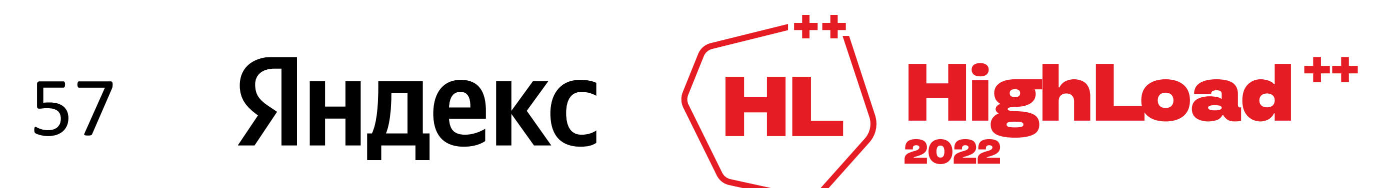


# Выводы

В большом хранилище:

+ Нужен явный контроль за записью (это не бесплатно)
+ Равномерное распределение шардов разных сервисов
+ Переезд шардов для выравнивания доли свободного места





# Выводы

В большом хранилище:

- Нужен явный контроль за записью (это не бесплатно)
- Равномерное распределение шардов разных сервисов
- Переезд шардов для выравнивания доли свободного места
- С помощью предсказания нагрузки можно улучшать балансировку и экономить место (= деньги)





# Спасибо!

**Вадим Зотеев**

Старший разработчик

vd.zoteev@yandex.ru

Слайды:
https://disk.yandex.ru/i/9L-NrdeYNX9WVg

HL HighLoad++ 2022

Яндекс

Голосуйте за мой доклад

